MULTIMEDIA UNIVERSITY

STUDENT ID NO

MULTIMEDIA UNIVERSITY

FINAL EXAMINATION

TRIMESTER 2, 2017/2018

ECE3166 ADVANCED MICROPROCESSORS

(All Majors)

6 MARCH 2018
9:00A.M. – 11:00A.M.
(2 Hours)

INSTRUCTIONS TO STUDENTS

1. This Question paper consists of NINE pages with FOUR Questions only.
2. Attempt **ALL** questions. All questions carry equal marks and the distribution of the marks for each question is given.
3. Please print all your answers in the Answer Booklet provided.

Question 1

(a) List and define **THREE** main characteristics that differentiate microprocessors. [3 marks]

(b) The architecture of ARM microprocessors is based on the Reduced Instruction Set Computer (RISC).

i. Compare and list **TWO** differences between complex instruction set computer (CISC) and RISC microprocessor systems. [4 marks]

ii. Three profiles have been defined in the ARM cortex series. List and describe the application areas of these **THREE** profiles. [3 marks]

(c) The 8086 microprocessor was Intel first 16-bit microprocessor.

i. State the size of data bus and address bus of the 8086 microprocessor in bits. [1 mark]

ii.. Determine the maximum size of memory address space and I/O address space in the 8086 microprocessor in bytes and list the lowest and highest addresses in the memory space and the I/O space in **hexadecimal**. [3 marks]

(d) The decimal data 19136220_{10} is stored in the memory of an Intel 8086 microprocessor starting from address 45670H in the little-endian byte ordering. Draw the data location in the memory and label the memory addresses and their corresponding contents in **hexadecimal**. [3 marks]

(e) Assume a CPU in 64-bit mode where the contents of registers are:

RAX = 0002 0001 8000 2201H

RBX = 0003 0003 0234 4402H

Show the contents of register RBX after the execution of the following assembly instructions.

Note: Each of the instructions execute on the **original values** of RAX and RBX given above. The instructions are **NOT** continuous.

i. ADD RBX, RAX

ii. ADD EBX, EAX

iii. ADD BX, AX

iv. ADD BL, AL

[8 marks]

Continued ...

Question 2

(a) The Intel 8086 microprocessor is the first member in the x86 family of microprocessor.

i. Name the **TWO** functional units in the 8086 microprocessor and describe their functions. [2 marks]

ii. There are three control flags in the status register of the 8086 microprocessor. Name any **TWO** of these control flags and explain their functions. [3 marks]

(b) If the initial contents of registers are as given in Table Q2(b), determine the addressing mode and physical address of the DESTINATION operand in each of the following instructions:

i. MOV ES:[1122H], AX [3 marks]

ii. MOV [BX][DI]+151H, CH [3 marks]

Table Q2(b): Contents of registers

Register	Contents	Register	Contents
AX	0101H	CS	1000H
BX	2903H	DS	2000H
CX	3561H	ES	3000H
DI	5AA5H	SS	4000H

Note: All workings must be shown.

(c) An x86 assembly instruction sequence is given in Table Q2(c). Complete and reproduce this table in your answer booklet. State the contents of the destination registers in **hexadecimal**. [6 marks]

Table Q2(c): An instruction sequence

Instruction	Contents of destination register
MOV AX, 1234H	AX = 1234H
MOV BX, 0ABCDH	BX = ABCDH
XOR AL, 5AH	
NOT BH	
SAR BL, 1	
ROL AH, 1	

(d) Write a real mode x86 instruction sequence to perform the following arithmetic operation and store the final result in register BX.

$$567_{10} - 37_{10} \times 21_{10}$$ [4 marks]

(e) A real mode x86 instruction sequence is given below:

```
MOV  SI, 5670H
MOV  DI, 7890H
MOV  CX, 3
CLD
REP  MOVSW
```

Initial contents of the segment registers are given as:

CS = 1000H, DS = 2000H, ES = 3000H. SS = 4000H

i. How many bytes of data will be moved in this instruction sequence? [1 mark]

ii. Analyze the instruction sequence and determine the physical addresses of the affected memory locations. [3 marks]

Continued ...

Question 3

(a) A data segment selector register is loaded with a value of 1147H. The base address of Global Descriptor Table (GDT) and Local Descriptor Table (LDT) are 0030 1100H and 0600 3200H, respectively.

[00302240H] = 00FFH	[06004340H] = 13FFH
[00302242H] = 0020H	[06004342H] = 0500H
[00302244H] = 9210H	[06004344H] = F392H
[00302246H] = 0500H	[06004346H] = 8100H

i. Find the Requested Privilege Level (RPL) and identify whether the descriptor is stored in the Global Descriptor Table (GDT) or Local Descriptor Table (LDT). [3 marks]

ii. Compute the address range of the data segment descriptor. [3 marks]

iii. Determine the base address and the size of the segment. [4 marks]

(b) Assume that 32-bit paging is enabled on an IA-32 processor with a page size of 4KB. The linear address of an operand is 22100820H, while the contents of the corresponding page directory entry and page table entry are 01206007H and 06002005H, respectively.

i. Compute the physical address range of the page table and page frame. [4 marks]

ii. Determine the physical address of the operand. [3 marks]

iii. Justify whether write access is allowed if the operand is accessed from user mode. [4 marks]

(c) Protection is a concept where memory is safeguarded against unauthorized access. Explain the code and data protection rule. [4 marks]

Continued ...

Question 4

(a) With the aid of suitable diagrams, briefly describe the differences between:

i. Early processors with one chip, one core and one executing thread.

ii. Processors with Intel Hyper-Threading Technology.

iii. Multi-core processors

[8 marks]

(b) Listing Q4 shows manipulation of single precision floating point numbers using SSE instructions.

```
struct SVector
{  float w, x, y, z;};
void SSE_Instr(struct SVector op_i, struct SVector op_j)
{  struct SVector ret_vector;
   int result = 0;
   __asm
      {      MOVUPS  XMM0, op_i   // Move the vectors to SSE regs
             MOVUPS  XMM1, op_j
             MOVUPS   XMM2, XMM0
             MAXPS    XMM3, XMM1
             MOVHLPS XMM2, XMM0
             MOVLHPS XMM2, XMM0
             SUBPS    XMM2, XMM1

      }      }
int main()
{   struct SVector obj1;   struct SVector obj2;    struct SVector res;
    obj1.w = 3.0; obj1.x = 7.0; obj1.y = 0.0; obj1.z = 6.0;
    obj2.w = 5.0; obj2.x = 1.0; obj2.y = 8.0; obj2.z = 2.0;
    SSE_Instr(obj1, obj2);    // Call the function, SSE_Instr.
      return 0;}
```

Listing Q4

Fill up and reproduce Table Q4 in your answer booklet to show the affected register and its content after execution of the following assembly lines. Show the contents in **hexadecimal.**

Table Q4

Instruction	Affected register = Content
MOVUPS XMM0, op_i	XMM0 = 40C000000000000040E0000040400000
MOVUPS XMM1, op_j	XMM1 = 400000004100000003F80000040A00000
MOVUPS XMM2, XMM0	
MAXPS XMM3, XMM1	
MOVHLPS XMM2, XMM0	
MOVLHPS XMM2, XMM0	
SUBPS XMM2, XMM1	

[13 marks]

(c) Discuss **TWO** advantages of 64-bit architecture over 32-bit architecture. [4 marks]

Continued ...

Appendix A

Instruction Set Summary for Intel 8086/80186/80286/80386/80486

AAA - Ascii Adjust for Addition
AAS - Ascii Adjust for Subtraction
ADC - Add With Carry
ADD - Arithmetic Addition
AND - Logical And
BSF - Bit Scan Forward (386+)
BSR - Bit Scan Reverse (386+)
BSWAP - Byte Swap (486+)
BT - Bit Test (386+)
BTC - Bit Test with Compliment (386+)
BTR - Bit Test with Reset (386+)
BTS - Bit Test and Set (386+)
CALL - Procedure Call
CBW - Convert Byte to Word
CDQ - Convert Double to Quad (386+)
CLC - Clear Carry
CLD - Clear Direction Flag
CLI - Clear Interrupt Flag (disable)
CLTS - Clear Task Switched Flag (286+ privileged)
CMC - Complement Carry Flag
CMP - Compare
CMPS - Compare String (Byte, Word or Doubleword)
CMPXCHG - Compare and Exchange
CWD - Convert Word to Doubleword
CWDE - Convert Word to Extended Doubleword (386+)
DAA - Decimal Adjust for Addition
DAS - Decimal Adjust for Subtraction
DEC - Decrement
DIV - Divide
ENTER - Make Stack Frame (80188+)
ESC - Escape
HLT - Halt CPU
IDIV - Signed Integer Division
IMUL - Signed Multiply
IN - Input Byte or Word From Port
INC - Increment
INT - Interrupt
INTO - Interrupt on Overflow
INVD - Invalidate Cache (486+)
INVLPG - Invalidate Translation Look-Aside Buffer Entry (486+)
IRET/IRETD - Interrupt Return
Jxx - Jump Instructions Table
JCXZ/JECXZ - Jump if Register (E)CX is Zero
JMP - Unconditional Jump
LAHF - Load Register AH From Flags
LAR - Load Access Rights (286+ protected)
LDS - Load Pointer Using DS
LEA - Load Effective Address
LES - Load Pointer Using ES
LFS - Load Pointer Using FS (386+)
LGDT - Load Global Descriptor Table (286+ privileged)
LIDT - Load Interrupt Descriptor Table (286+ privileged)
LGS - Load Pointer Using GS (386+)
LLDT - Load Local Descriptor Table (286+ privileged)
LMSW - Load Machine Status Word (286+ privileged)

Continued ...

LOCK - Lock Bus
LODS - Load String (Byte, Word or Double)
LOOP - Decrement CX and Loop if CX Not Zero
LOOPE/LOOPZ - Loop While Equal / Loop While Zero
LOOPNZ/LOOPNE - Loop While Not Zero / Loop While Not Equal
LSL - Load Segment Limit (286+ protected)
LSS - Load Pointer Using SS (386+)
LTR - Load Task Register (286+ privileged)
MOV - Move Byte or Word
MOVS - Move String (Byte or Word)
MOVSX - Move with Sign Extend (386+)
MOVZX - Move with Zero Extend (386+)
MUL - Unsigned Multiply
NEG - Two's Complement Negation
NOP - No Operation (90h)
OR - Inclusive Logical OR
OUT - Output Data to Port
POP - Pop Word off Stack
POPA/POPAD - Pop All Registers onto Stack (80188+)
POPF/POPFD - Pop Flags off Stack
PUSH - Push Word onto Stack
PUSHA/PUSHAD - Push All Registers onto Stack (80188+)
PUSHF/PUSHFD - Push Flags onto Stack
RCL - Rotate Through Carry Left
RCR - Rotate Through Carry Right
REP - Repeat String Operation
REPE/REPZ - Repeat Equal / Repeat Zero
REPNE/REPNZ - Repeat Not Equal / Repeat Not Zero
RET/RETF - Return From Procedure
ROL - Rotate Left
ROR - Rotate Right
SAHF - Store AH Register into FLAGS
SAL/SHL - Shift Arithmetic Left / Shift Logical Left
SAR - Shift Arithmetic Right
SBB - Subtract with Borrow/Carry
SCAS - Scan String (Byte, Word or Doubleword)
SETS - Set if Signed (386+)
SETNS - Set if Not Signed (386+)
SETC - Set if Carry (386+)
SETNC - Set if Not Carry (386+)
SETO - Set if Overflow (386+)
SETP/SETPE - Set if Parity / Set if Parity Even (386+)
SETNP/SETPO - Set if No Parity / Set if Parity Odd (386+)
SGDT - Store Global Descriptor Table (286+ privileged)
SIDT - Store Interrupt Descriptor Table (286+ privileged)
SHL - Shift Logical Left
SHR - Shift Logical Right
SHLD/SHRD - Double Precision Shift (386+)
SLDT - Store Local Descriptor Table (286+ privileged)
SMSW - Store Machine Status Word (286+ privileged)
STC - Set Carry
STD - Set Direction Flag
STI - Set Interrupt Flag (Enable Interrupts)
STOS - Store String (Byte, Word or Doubleword)
STR - Store Task Register (286+ privileged)
SUB - Subtract
TEST - Test for Bit Pattern
WBINVD - Write-Back and Invalidate Cache (486+)
XCHG - Exchange
XOR - Exclusive OR

Continued ...

Mnemonic	Meaning	Format	Operation	Flags Affected
MOVS	Move string	MOVSB/MOVSW	((ES)0 + (DI)) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	None
CMPS	Compare string	CMPSB/CMPSW	Set flags as per ((DS)0 + (SI)) − ((ES)0 + (DI)) (SI) ← (SI) ± 1 or 2 (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
SCAS	Scan string	SCASB/SCASW	Set flags as per (AL or AX) − ((ES)0 + (DI)) (DI) ← (DI) ± 1 or 2	CF, PF, AF, ZF, SF, OF
LODS	Load string	LODSB/LODSW	(AL or AX) ← ((DS)0 + (SI)) (SI) ← (SI) ± 1 or 2	None
STOS	Store string	STOSB/STOSW	((ES)0 + (DI)) ← (AL or AX) (DI) ← (DI) ± 1 or 2	None

Prefix	Used with:	Meaning
REP	MOVS STOS	Repeat while not end of string CX ≠ 0
REPE/REPZ	CMPS SCAS	Repeat while not end of string and strings are equal CX ≠ 0 and ZF = 1
REPNE/REPNZ	CMPS SCAS	Repeat while not end of string and strings are not equal CX ≠ 0 and ZF = 0

Appendix B

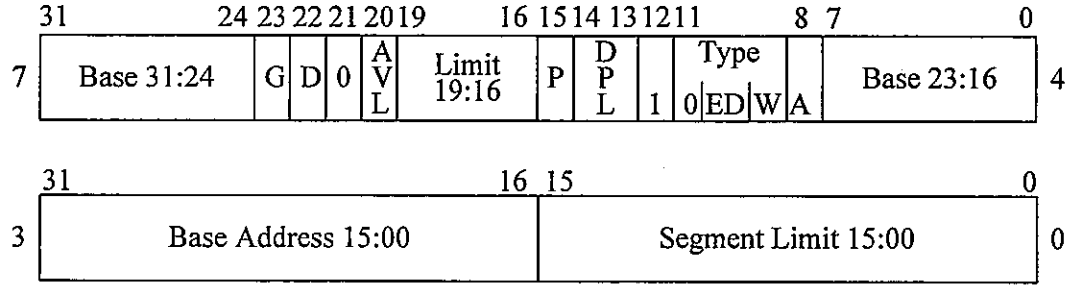
Summary of Selected SSE Instructions

Mnemonic	Description
MOVAPS	Move aligned packed single-precision floating-point values
MOVUPS	Move unaligned packed single-precision floating-point values
MOVHPS	Move high packed single-precision floating-point values
MOVHLPS	Move packed single-precision floating-point high to low
MOVLHPS	Move packed single-precision floating-point low to high
MOVLPS	Move low packed single-precision floating-point values
MOVMSKPS	Move packed single-precision floating-point mask
MOVSS	Move scalar single-precision floating-point values
ADDSS	Scalar add pair of operands
ADDPS	Packed add pairs of operands
SUBSS	Scalar subtract pair of operands
SUBPS	Packed subtract pair of operands
MINPS	Packed return the maximum of the pair of operands
MAXPS	Packed return the minimum of the pair of operands

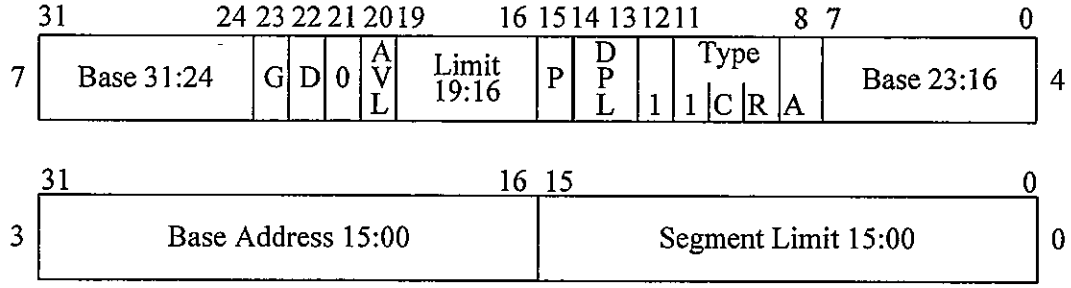
Continued...

Appendix C

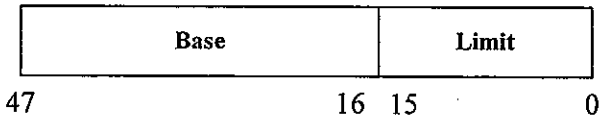
Data-Segment Descriptor:



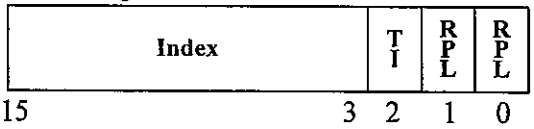
Code-Segment Descriptor:



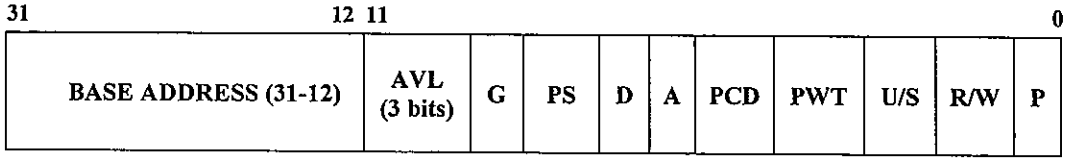
Global Descriptor Table Register (GDTR) and Interrupt Descriptor Table Register (IDTR):



Selector Register:



Page Directory or Page Table Entry Format:



End of Paper